A brief guide to PRISM

Taisuke Sato and Yoshitaka Kameya

PRISM was proposed in 1997 as a probabilistic extension of Prolog. It inherits from Prolog the syntax and the least model semantics, with the latter being generalized to the distribution semantics, i.e. possible worlds with a distribution (probability measure). It was a first logic-based programming language for probabilistic modeling with a mechanism of parameter (probability) learning.

Currently PRISM, version 2.0 built on top of B-Prolog, offers a variety of functionalities for machine leaning tasks. We understand that probabilistic modeling is similar to programming; it is not a one-shot process but goes through cycles of building, tuning and debugging models. PRISM makes this long painful process short, first by offering high level logical expressions that make your programs short, and second by making available many generic built-ins for machine learning tasks that eliminate the need of inventing and implementing new algorithms for your models. You can do the following with PRISM (items with $*$ will be available soon).

[**Model construction**] You can build probabilistic models by writing PRISM programs just like writing Prolog programs. They are generative from the viewpoint of machine learning since a program describes how an observation (observable atom) is probabilistically generated using `msw` atoms representing multi-ary random switches. Generative models include BNs (Bayesian networks) and probabilistic grammars up to type-0. Programs have to satisfy certain conditions to correctly specify distributions.

[**Sampling**]

- **Forward sampling:** Samples are drawn from the distribution defined by a program.
- **MCMC sampling*:** Bayesian inference based on Metropolis-Hastings style MCMC with Dirichlet priors.

[**Probabilistic inference**]

- **Exact inference:** Probabilities $P(G)$ of goals G and conditional probabilities $P(G'|G)$ where G' is a subgoal appearing in a proof of G are computed efficiently by dynamic programming. They correspond to marginal distribution and conditional distribution respectively. PRISM's probability computation subsumes various well-known machine learning algorithms such as the junction tree algorithm for BNs, the Baum-Welch algorithm for HMMs (hidden Markov models) and the Inside-outside algorithm for PCFGs (probabilistic context free grammars).

- **Viterbi inference:** The most likely instantiation for the given goal is computed together with its derivation path (i.e. explanation in the sense of abduction) and probability.

[**Statistical learning**]

- **Parameter learning:** Parameters are learned from data by **MLE** (maximum likelihood estimation) and **MAP** (maximum a posteriori estimation) by the EM algorithm. Also **DAEM** (deterministic annealing EM) is available to avoid local optima.
- **Viterbi learning*:** This is an approximate parameter learning aimed at scalability and speed up. An order of magnitude speed up is observed in the case of a preliminary experiment with HMMs.
- **VB (variational Bayes) inference:** Approximate Bayesian inference with Dirichlet priors over `msw` atoms for posterior distributions.

[**Model selection**] **BIC** (Bayesian information criterion), **CS** (Cheeseman-Stutz) criterion and **VFE** (variational free energy) are automatically computed for models and data. These are quite useful for structure learning.

PRISM has constantly been improved for more than ten years and the code now seems stable and reliable. For example, you can safely use up 64 gigabytes memory for parameter learning of PCFGs without memory trouble. Also there are lots of convenient built-in predicates for machine learning experiments from loop macros to ones for computing standard statistics. The PRISM package is downloadable at `http://sato-www.cs.titech.ac.jp/prism/`. The user can use this package free of charge for academic use and also can modify the source code under the modified BSD license.